UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/552,135	04/19/2000	Naoki Oguchi	FUJZ 17.260	3482

26304 7590 06/07/2004

KATTEN MUCHIN ZAVIS ROSENMAN
575 MADISON AVENUE
NEW YORK, NY 10022-2585

EXAMINER	
HO, CHUONG T	
ART UNIT	**PAPER NUMBER**
2664	7

DATE MAILED: 06/07/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)
	09/552,135	OGUCHI, NAOKI
	Examiner	**Art Unit**
	Chuong Ho	2664

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 March 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-13* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-13* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

1. The amendment filed 03/10/04 have been entered and made of record.

2. Applicant's arguments with respect to claims 1-13 have been considered but are moot in view of the new ground(s) of rejection.

3. Claims 1-13 are pending.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) and the Intellectual Property and High Technology Technical Amendments Act of 2002 do not apply when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. Therefore, the prior art date of the reference is determined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

5. Claims 1-13 are rejected under 35 U.S.C. 102(e) as being anticipated by Muller et al. (U.S.Patent No. 6,480,489 B1) (Filed March 01, 1999).

In the claim 1, Muller et al. discloses if the packet was formatted with one of the set of predetermined protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g.,

how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24); comprising:

. notifying a free space of the receiving buffer (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

reassembly a plurality of receiving packets into a single big packet, based on the free space, to be transmitted to the receiving buffer (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may

bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

determining a size of the big packet based on the free space (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52) if the packet was formatted with one of the set of predetermined protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage

space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

receiving buffer of an upper layer (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52)).

In the claim 2, Muller et al. discloses the packet processing device wherein the first means is included in the upper layer and notifies the free space to the third (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

6. determining a size of the big packet based on the free space (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52) if the packet was formatted with one of the set of predetermined

protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24)).

In the claim 3, Muller et al. discloses a backward packet inclusive information reading circuit detecting the free space based on information within a backward packet from the upper layer (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely

populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

7. determining a size of the big packet based on the free space (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52) if the packet was formatted with one of the set of predetermined protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24)).

In the claim 4, Muller et al. discloses the upper layer comprises a transport layer (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then,

the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

8. determining a size of the big packet based on the free space (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52) if the packet was formatted with one of the set of predetermined protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage

space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24)).

In the claim 5, Muller et al. discloses the upper layer comprises an application layer and the big packet is transmitted not through a buffer of transport layer but directly to the receiving buffer (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

9. determining a size of the big packet based on the free space (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52) if the packet was formatted with one of the set of predetermined protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have

been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24)).

In the claim 6, Muller et al. discloses identifying a connection of the receiving packets, reassembling the big packet for each connection based on identification information (The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24));

10. determining a size of the big packet based on the free space (the flow re-assembly buffer may be used to re-assemble data from multiple packets of a single

communication flow. Collecting multiple data portion in a single buffer, illustratively of memory page size, allows efficient transfer of the data to a destination application (see col. 4, lines 48-52) if the packet was formatted with one of the set of predetermined protocol, its data is re-assembly in re-assembly buffer with data from other packet in the same communication flow (See abstract). The flow re-assembly buffer is examined to determine if it is full...by first determining how much data (e.g., how may bytes) have been stored in the buffer. Illustratively, the flow's next address field and amount of data stored from this packet are summed. Then, the initial buffer address (e.g., before any data was stored in it) is subtracted from this sum. This value, representing how much data is now stored in the buffer, is then compared to the size of the buffer (e.g., eight kilobytes). If the amount of data currently stored in the buffer equals the size of the buffer, the it is full...Thus, a flow re-assembly buffer is not considered full until it storage space is completely populated with flow data. This scheme enables the efficient processing of network packets (see col. 86, lines 10-24)).

11. In the claim 7, Muller et al. discloses a checksum calculating circuit adding a checksum to the big packet (see col. 11, lines 9-14).

12. In the claim 8, Muller et al. discloses a timer for giving the instructions for transmitting the big packet to the receiving buffer when a predetermined time elapses (aww xol. 112, lines 32-40).

13. In the claim 9, Muller et al. discloses assigning the big packet to the receiving buffer at a time when the big packet attains a size for issuance of an acknowledgement packet from the upper layer (see col.105, lines 53-56).

14. In the claim 10, Muller et al. discloses assembling the big packet with a first receiving packet including a header and subsequently received packets whose headers are deleted (see col. 80, lines 10-15).

15. In the claim 11, Muller et al. discloses transmitting the receiving packet to the receiving buffer without storing the receiving packet in the second means when the receiving packet is a non-accumulation packet (non re-assembly packet) (see col. 57, lines 10-25).

16. In the claim 12, Muller et al. discloses a packet transfer function in a network layer made in a hardware form and transmits a plurality of receiving packets address to itself to the second means (see col. 59, lines 50-55).

17. In the claim 13, Muller et al. discloses an NIC (network interface card) device comprising the packet processing device which transmits a plurality of receiving packets addressed to itself to the second means (see col. 7, lines 38-45, col. 59, lines 50-55).

Consclusion

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chuong Ho whose telephone number is (703) 306-4529. The examiner can normally be reached on 8:00AM to 4:00PM.

19. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chuong Ho
Examiner
Art Unit 2664

05/25/04

WELLINGTON CHIN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2600